UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/608,985	06/26/2003	Amitabh Srivastava	3382-64710	6401

26119 7590 10/04/2007

KLARQUIST SPARKMAN LLP
121 S.W. SALMON STREET
SUITE 1600
PORTLAND, OR 97204

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
10/04/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/608,985	SRIVASTAVA ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 September 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-36* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-36* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *9/13/07*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 9/13/07.

As indicated in Applicant's response, claims 1, 21, 27, 29, 32 have been amended.

Claims 1-36 are pending in the office action.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 21, 25, 29 provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 1, 14 of copending Application No. 11,330,053 (hereinafter '053), further in view of Srivastava et al., "Effectively Prioritizing Tests in Development Environment", February 2002, MSR-TR-2002-15, Publisher: *Association for Computing Machinery, Inc* (hereinafter Srivastava). Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following conflicts.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

As per instant claims 21, 25, and 29, '053 claims 1, 14 also recite prioritizing test (which include performing tests) based on indicators indicating which portions are impacted by tests, portions including plurality of blocks wherein indication (i) indicates ones of the plurality of blocks are modified, hence this reads on marking changed logical abstractions; but '503 does not recite:

determining dependency information about binary files, propagating such information to determine subsystem and system dependency, marking changed and unchanged logical abstraction to prioritize tests.

Srivastava in a environment for obtaining list prioritized tests (see pg. 5-7) to perform tests, similar to '053; and not only discloses using previous test results or data being marked to feed into new prioritized test, but also discloses binary files analysis to block system and subsystem, recording changes thereto in a propagation analysis hashing (L column, 4th para, pg. 3), and further obtaining dependency information (e.g. *along with ... coverage information* – L column 1st para, pg. 2; *changes ... propagated* – R column, 5th para, pg. 2; *list of modified blocks* – top para, R column. pg. 3; *marked as old block, impacted blocks, matching blocks* -- pg. 3, R column) so as to propagate it into graph elements in order to mark intra-procedural or interprocedural changes so to determine what are new block or unchanged blocks or potential impacted new block in view of a subsequent test coverage (see Fig. 1, Fig. 2, pg. 3-4); and producing prioritized lists (see top pg. 3).

It would have been obvious for one skill in the art to implement the prioritized tests by instant claims 21, 25, and 29 so that '053 steps of creating list of prioritized tests recording modified or new blocks with respect to a given test version, which is analogous to Srivastava from above, be implemented to include using test results into a test optimization instance and effecting for which a dependency information determination and propagation via graph and hashing algorithm (see Srivastava) to enable marking of system and subsystem of abstractions in binary files as taught by Srivastava, because this enable appropriate marking as suggested in (i) leading to optimization on how to reduce paths in recurring tests for a given test version based on dependency information propagated in target binary files as Srivastava discloses (see Fig. 2).

Claim Objections

4. Claim 18 is objected for the following informality: 'number of ... abstractions effected by the proposed change...'; and the proper term in the above context for 'effected by' and in light of the Specifications should be 'affected by'. Appropriate correction is required.

Claim Rejections - 35 USC § 112

5. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. Claims 1-20, 27-28 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Specifically, claim 1 recites 'receiving, responsive the requests sent via the application programming interface, a dependency collection'. There insufficient antecedent basis for the

plural term 'requests' since the claim only mentions about a dependency request comprised in a API implementing a system dependency creation request; and accordingly 'requests' will be treated as said 'system dependency creation request'.

Claims 2-20, for not clarifying on the 'requests' limitation, are also rejected for the same reason.

Claim 27 also recites 'responsive to the requests sent via the application program inteface'; hence is rejected along with claim 28 for lack of antecedent basis.

Claim 15 recites 'wherein the further comprising' and this typographical impropriety cannot enable one of ordinary skill in the art how to ascertain what exact element is being described; hence will be treated as claim 1 further comprising.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. Claims 1-16, 20, 27-28, 32, and 36 are rejected under 35 U.S.C. 102(b) as being anticipated by Srivastava et al, "Vulcan: Binary transformation in a distributed environment", April 2001, Technical Report MSR-TR-2001-50, pp. 1-12 (hereinafter Srivastava).

As per claim 1, Srivastava discloses a method comprising:

receiving a system definition (e.g. *PDB file, meta-information* – sec 3.1,pg. 6, R col) and a request for dependency information (Fig. 2 – Note: submission of input into a API reads on

request using Vulcan), the system definition comprising metadata describing a system at a level of abstraction (e.g. *PDB file, meta-information* – sec 3.1,pg. 6, R col to pg. 7, top L col);

requesting, via an application programming interface, a dependency collection, by sending to the application programming interface a system dependency creation request (e.g. Fig. 2) comprising the received system definition and a dependency request (sec 2.1 pg. 5 – Note: input form of abstraction submitted in a API to yield output level via externalizing – see Fig. 2 -- the received abstractions; *Vulcan API* - Appendix, pg. 11 discloses *via a application programming interface*) comprising a target logical abstraction;

receiving, responsive to the requests sent via the application programming interface, a dependency collection (Fig. 3; *Vulcan API, chaining* – Appendix A, pg. 11) for the target logical abstraction comprising logical abstractions in one or more dependency chains with the target logical abstraction (e.g. *six basic abstractions, list of Basic Block, CFG, DFG* – sec. 2.2 pg. 5, R col),

wherein the dependency collection comprises logical abstractions outside of a subsystem for the target logical abstraction (see *abstraction ...different component ...heterogeneous* – pg. 4, R col; Fig. 1); and

displaying a representation of the collection (e.g. *externalized ...for... transformation, testing debugging* – pg. 4, R col. bottom; *abstract representation .. queried analysis modified for instrumentation* – sec 2.1, pg. 5 L col – Note: Win32, Microsoft system and IA 64 architecture based tool with externalization of abstraction for debugging, modification and optimizing reads on displaying).

As per claims 2-3, Srivastava discloses wherein the system definition is received as a file identification (e.g. *meta-information ... PDB file* – sec 3.1, pg. 6, R col); wherein the system definition is received via a graphical user interface (refer to Vulcan debug tool with display as per claim 1)

As per claims 4 and 7, Srivastava discloses wherein the target logical abstraction is an unchanged logical abstraction (e.g. *Basic Block, CFG, DFG* – sec 2.2. pg. 5 R col.), wherein the target logical abstraction is a changed logical abstraction (e.g. *instrumentation interface ...inserted ... Basic Block* – pg. 6, L top col.).

As per claims 5-6, Srivastava discloses wherein the dependency collection comprises logical abstractions dependent on (Fig. 2; sec 2.1, pg. 5) the target logical abstraction; wherein the dependency collection comprises logical abstractions (e.g. Fig. 2; Intel x86, Risc, Compaq Alpha – pg. 5, R col.) on which the target logical abstraction depends.

As per claims 8-9, Srivastava discloses wherein the target logical abstraction comprises a basic block, a procedure, or a binary file (Procedure, Basic block, sec 2.2, pg. 5); wherein the dependency collection comprises a basic block logical abstraction (e.g. pg 5, R col).

As per claim 10, refer to claim 8, wherein the dependency collection comprises at least one of a procedure logical abstraction or a binary file logical abstraction.

As per claim 11, Srivastava discloses wherein the dependency collection comprises a named object logical abstraction (e.g. *Component* – pg. 5, top R col) or a node logical abstraction (e.g. CFG, DFG - pg. 5, R col.).

As per claim 12, Srivastava discloses wherein the representation of the collection comprises a number of affected logical abstractions (e.g. instrumentation …inserts probes … reorders basic blocks – Fig. 6, pg. 7 L col. top).

As per claim 13-15, Srivastava discloses wherein the representation of the collection comprises a graphical presentation of a dependency chain (graphical display: refer to claim 1; *Basic Block, CFG, DFG* – sec 2.2. pg. 5 R col) wherein the representation of the collection comprises a list of logical abstractions (e.g. *list of Basic Blocks* -pg. 5, R col); and further comprising displaying a representation of the collection comprises a graph of logical abstractions (re claim 1).

As per claim 16, Srivastava discloses wherein the dependency collection further comprises logical abstractions (e.g. *System, collection of Programs, list of components, Component consisting ... symbol, data blocks* – pg. 5, top R col) inside the logical abstraction's subsystem.

As per claim 20, refer to claim for computer-readable medium having executable instructions for performing the method of claim 1.

As per claim 27, Srivastava discloses a computer-readable medium (see claim 20) having executable instructions performing a method comprising:

creating a system definition comprising metadata describing a system at a level of abstraction in response to receiving graphical user interface input;

receiving a dependency information request via graphical user interface input;

requesting via an application programming interlace exposed by a dependency framework, a dependency collection, by sending to the application programming interface a

system dependency creation request comprising the system definition, and a target logical abstraction identifiable from the dependency information request; and

receiving, responsive to the requests sent via the application programming interface, a dependency collection for the target logical abstraction comprising logical abstractions in one or more dependency chains with the target logical abstraction;

wherein the dependency collection comprises logical abstractions outside of a subsystem for the logical abstraction;

all of which limitations having been addressed in claim 1.

As per claim 28, refer to claim 16.

As per claim 32, Srivastava discloses a user interface service comprising:

means for accepting a system definition comprising metadata (*PDB file, meta-information* – sec 3.1,pg. 6, R col) describing a system at a level of abstraction (sec 2.2 pg. 5) and indicating binary files (e.g. *from the application binaries* - sec 2.1 pg. 5) in plural subsystems;

means for accepting an indication (sec 2.2, pg. 5; Fig. 2 – Note: input into API for requesting abstraction representation by Vulcan reads on indication) of a target logical abstraction; and

means for displaying dependency relationships (e.g. *externalized ...for... transformation, testing debugging* – pg. 4, R col. bottom; *abstract representation .. queried analysis modified for instrumentation* – sec 2.1, pg. 5 L col – Note: Win32, Microsoft system and IA 64 architecture based tool with externalization of abstraction for debugging, modification and optimizing reads

on displaying) between the target logical abstraction and a set of logical abstractions in binary files from two or more of the plural subsystems.

As per claim 36, Srivastava discloses means for displaying test coverage evaluation (pg. results (e.g. *code coverage ... test* – pg. 9, R col. top; Fig. 7-10).

9. Claims 21-26, 29-31 are rejected under 35 U.S.C. 102(b) as being anticipated by Srivastava et al., "Effectively Prioritizing Tests in Development Environment", February 2002, MSR-TR-2002-15, Publisher: *Association for Computing Machinery, Inc.*, pp. 1-10 (hereinafter Srivastava_2).

As per claim 21, Srivastava_2 discloses method comprising:

determining dependency information about binary files (e.g. *along with ... coverage information* – L column 1st para, pg. 2; *changes ... propagated* – R column, 5th para, pg. 2; *list of modified blocks* – top para, R column. pg. 3; *marked as old block, impacted blocks, matching blocks* -- pg. 3, R column - Note: using previous results or changes in files, and determining blocks for marking and marking insides binary files to find out about subsequent path on how the block test coverage is to be done reads on determining dependency information based on which to apply optimization algorithm upon runtime resources);

propagating dependency information about binary files to determine subsystem dependency information for a subsystem of the system; propagating the subsystem dependency information to determine system dependency information for the system (e.g. *changes ... propagated* – R column, 5th para, pg. 2 – Note: marking from using information changes to affected parts of a source binary as this is scanned under analysis reads on propagating this

information determination or change information and use it for marking of blocks for a given file version, i.e. subsystem of system or dependency of system – see Table 1, Table 2, pg. 4 in light of algorithm of Fig. 2 where iterative sub-block are included into higher blocks for a coverage);

marking changed logical abstractions; marking unchanged logical abstractions dependent on marked changed logical abstractions in other subsystems (*marked as old block, impacted blocks, matching blocks* -- pg. 3 R column);

comparing test coverage to marked changed logical abstractions and to marked unchanged logical abstractions (e.g. *heuristic ...successor predecessor blocks* – pg. 3, R column ; *likely to be taken* – R column, bottom pg. 3; Fig. 2 – Note: based on marking information between old and new block, and applying heuristic thereto in order to predict how to skip path reads on comparing based on marked of changed and unchanged); and

prioritizing tests based on maximum test coverage of marked changed logical abstractions and marked unchanged logical abstractions (e.g. Fig. 2);

and performing the prioritized tests according to test priorities to produce test results (top para, L col., pg. 2; Figs. 8, 10, pg. 6-7).

As per claims 22-23, Srivastava_2 discloses coverage comprises tests for one subsystem (e.g. sequence Set – Fig 2 – Note: block set being tested based on weight of impacted blocks reads on subsystem of binary blocks), a subsystem among a plural subsystems.

As per claim 24, Srivastava_2 discloses the test coverage comprises tests for plural subsystems and maximum test coverage is considered for marked changed logical abstractions and marked unchanged logical abstractions (pg. 4, L column; Fig. 2 – Note: impacted blocks

based on marking of old and new blocks reads on test coverage for marked changed and unchanged – see R col., pg. 3) for said plural subsystems.

As per claim 25, Srivastava_2 discloses a computer-based service comprising **means for** performing the same steps as recited in claim 21, namely:

determining binary dependencies for a defined system (e.g. *along with ... coverage information* – L column 1st para, pg. 2; *changes ... propagated* – R column, 5th para, pg. 2; *marked as old block, impacted blocks, matching blocks* -- pg. 3 R column);

propagating binary dependencies to identify binaries dependent on binaries in other subsystems and storing determined and propagated dependencies (*changes ... propagated* – R column, 5th para, pg. 2 - Note: marking from using information changes to affected parts of a source binary as this is scanned under analysis reads on propagating this information determination or change information and use it for marking);

marking changes and propagating marked changes using the determined and propagated dependencies; and prioritizing tests based on test coverage of marked changes and propagated marked changes;

performing prioritized tests according to the prioritizing;

all of which steps having been addressed in claim 21.

As per claim 26, Srivastava_2 discloses marking proposed changes (e.g. *proposed* – top para, R column, pg. 2; *new version ... likely to be covered by a existing test* – 3rd para, R column, pg. 3 – Note: applying a test to a new code reads on proposed changes to an current test scenario).

As per claim 29, Srivastava_2 discloses computer system comprising:

a processor coupled to volatile and nonvolatile memory; binary files stored in memory (see Binary - Fig. 1); software stored in memory comprising computer executable instructions for:

determining dependency information for binary files, propagating dependency information to determine subsystem dependency information for a subsystem of a system, and propagating subsystem dependency information to determine system dependency information for the system (refer to claim 21);

marking logical abstractions changed from a previous version (e.g. *Old Binary, New Binary* - Fig .1, R column pg. 3);

propagating marked changes according to the dependency information comprising marking unchanged logical abstractions dependent on marked changes in other subsystems (refer to claim 21);

comparing test coverage to marked changed logical abstractions and to marked unchanged logical abstractions; and prioritizing tests based on maximum test coverage of marked changed logical abstractions and marked unchanged logical abstractions (refer to corresponding rejection in claim 21).

As per claims 30-31, Srivastava_2 discloses maximum test coverage is based on the total number of marked changed and marked unchanged logical abstractions touched (e.g. Fig. 2; L column, pg. 4) by a test system wide (Note: covering of subset or test sequence per iteration reads on system wide); wherein maximum test coverage is based on the sum of the total number of marked changed logical abstractions in a first subsystem touched by a test, and marked unchanged logical abstractions touched by the test in the first subsystem, wherein the marked

unchanged logical abstractions depend on marked changed logical abstractions in other subsystems (e.g. pg. 3, R column 2nd para, bottom para, pg. 3 to L column, pg. 4; ch. 4-5, pg. 4-5).

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 17-19, 33-35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Srivastava et al, "Vulcan: Binary transformation in a distributed environment", April 2001, Technical Report MSR-TR-2001-50 (Srivastava), in view of Srivastava et al., "Effectively Prioritizing Tests in Development Environment", February 2002, MSR-TR-2002-15, Publisher: *Association for Computing Machinery, Inc.*, pp. 1-10 (Srivastava_2).

As per claims 17-18, Srivastava does not explicitly disclose wherein the logical abstraction comprise a proposed change, and displaying a metric for said proposed change risk, wherein said risk comprise a number of abstractions effected by the proposed change in relation to the number of abstractions. In a framework analogous to Srivastava (see Srivastava: sec 5 pg 9) for analyzing possible changes to binary files across systems including, instrumentation, recompiling to detect risks from replaying previous version thereof and identifying stale profile, Srivastava_2 discloses using Vulcan in conjunction with BMAT (as Srivastava) and further teaches code execution for detecting impacted of block of codes in view of the number of blocks

being observed (e.g. Fig. 4-5, pg. 5 – Note: percentage of impacted blocks over total sequences reads on display of proposed risk changes in view of proposed changes - *proposed* – top para, R column, pg. 2); i.e. percentage reads on displaying a change risk metric. Hence, in view of the same environment using instrumentation/profiling for binary changes analysis applicable to multi-system application version improvement, it would have been obvious for one skill in the art at the time the invention was made to enable the abstraction representation in Srivastava's debugging and dynamic instrumentation tool such that it include the analysis of binary code as in Srivastava_2 so that a impact percentage among the level of abstraction being analyzed by the tool be marked as a displayed metric so to support whether the whole code incur sufficient risk so that the software development team can decide to rebuild the target program binary code based on such percentage metric as endeavored by Srivastava_2's large scale prioritization of applications and (see Srivastava_2: Abstract, Introduction, pg. 1;), which is commonly contemplated by Srivastava for scalability within larger commercial application including optimizing via determination on how much to recompile (see Introduction, pg. 3; Partial compilation -sec 6.1, pg. 9).

As per claim 19, Srivastava_2 discloses wherein the relation is further adapted with a logarithmic function (see Fig. 5, pg. 5); hence the rationale so to display a metric within such logarithmic function of binary code behavior would have been obvious in view of the same tool and endeavor by Srivastava and Srivastava_2.

As per claims 33-34, Srivastava's tool display is not disclosed as comprising means for displaying a proposed change risk and means for displaying a change risk metric. But these limitations have been addressed in the rationale set forth for claims 17-18 respectively.

As per claim 35, Srivastava's tool and user interface is not disclosed as comprising means for displaying a graph of relative risk for plural subsystems. But the graph with metrics indicating a percentage of impact leading to evaluation of risk and subsequent determination for rebuilding binary has been addressed in claims 17-18, as obvious in view of the similar endeavor and purports by both Srivastava and Srivastava_2.

Response to Arguments

12. Applicant's arguments filed 9/13/07 have been fully considered but they are either mostly moot, or not persuasive. Following are the Examiner's observations in regard thereto.

35 USC § 102 Rejection under Srivastava_2:

(A) For claim 21: Applicants have submitted that Srivastava (now Srivastava_2) only propagates changes found in binary files, not dependency information about binary files … to determine system dependency information' because Srivastava has the binary files with propagation already done (Appl. Rmrks pg. 12, middle). The rejection of claim 21 has provided distinct citations with respect to addressing the features of claim 1, that is:

> (e.g. *along with … coverage information* – L column 1st para, pg. 2; *changes … propagated* – R column, 5th para, pg. 2; *list of modified blocks* – top para, R column. pg. 3; *marked as old block, impacted blocks, matching blocks* -- pg. 3, R column - Note: using previous results or changes in files, and determining blocks for marking and marking insides binary files to find out about subsequent path on how the block test coverage is to be done reads on determining dependency information based on which to apply optimization algorithm upon runtime resources)
> and,
> (e.g. *changes … propagated* – R column, 5th para, pg. 2 – Note: marking from using information changes to affected parts of a source binary as this is scanned under analysis reads on propagating this information determination or change information and use it for marking of blocks for a given file version, i.e. subsystem of system or dependency of system – see Table 1, Table 2, pg. 4 in light of algorithm of Fig. 2 where iterative sub-block are included into higher blocks for a coverage)

That is, previous marking leading to determination of what to readjust for test prioritizing, and effectuation code modifications represents a sequence of determinating steps leading to other sequences of binary data identification leading to re-ordering algorithmic step implementation/instrumentation performed upon the block hierarchy (of the binary blocks) being observed, so that the final modified code would reflect such amount of modifications. The above sequence reads on the act of propagating; and the 'subsystem dependency information' is analogized to the very nature of dependency of binary blocks that are laid out as a graph by Echelon; that is, a lay-out where sub-blocks within the scope of larger blocks are analyzed based on the above marking and the contextual block relationship that would be based upon for readjusting (see Fig. 1, Fig. 2 of Srivastava) and re-instrumenting code modification for their prioritization test. The claim does not describe any particular implementation detail that would preclude the sequential analysis and instrumentation steps above from reading on the term 'propagating' and 'system dependency'. As for data propagation, this very act of 'propagating' has to entail flow of data going from one point to another point. The claim not only does not specify what type of data is flowing nor does it include existence of a node-to-node binary structure, but also is silent about the nature of the location or endpoints where such data is supposed to reach or moved to. Broad interpretation has been used and the above analysis has clarified how the language of the claim has been mapped with the algorithm working on blocks graph and analysis sequence used by Srivastava_2. In prosecuting a case, it is deemed that merits would be properly given to a clearly conveyed limitation within a claimed scenario, that in a whole, yields sufficient teaching enabling one to construe the metes and bounds of the invention. The phrase 'propagating ... information about binary files ... to determine ...

dependency information' appears to be truncated semantic or a omission of essential teachings -- about this *propagating* and *determining* steps; because how reasonably and substantially explicit and informative a limitation is laid out would contribute in determining any patentability weight. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

(B) Applicants argue that the cited portions in Srivastava merely teach that what appears to be 'propagated' is changes not 'dependency' (Appl. Rmrks pg. 12 bottom). The nature of code being configured as blocks by Echelon necessarily contains dependency and by identifying the markings at specific points of the graph of binary blocks, the very step of identifying entails how a proposed change to a spot in such lay-out would be intrinsically affected by another part of the graph; so dependency is **integral** to the whole process of operating upon the points at which to provide code changes and based thereupon make proper reordering; i.e. reordering and recompiling a binary laid out cannot be done without dependency knowledge among such lay-out when parts of the lay-out have been analyzed throughout the analysis sequence (as set forth in section A) using Echelon for code reprioritizing.

(C) Applicants have submitted that the Fig. 2 and Tables 1-2 proffered by the Office Action cannot be mapped with what is recited as 'propagation' (Appl. Rmrks pg. 13, top). The 'propagating' limitation has been interpreted and met by the Rejection as explained in section A above. The argument is non-persuasive.

(D) As for claims 25 and 29, the Applicants' argument would have to be referred to the reply sections set forth above.

35 USC § 102 Rejections of claims 1, 27, 32:

(E) For the rest of the arguments, these are moot in light of the new grounds of rejection.

Double Patenting Rejection (Appl. Rmrks, pg. 15):

(F) As long as the claim is not convincingly definite in regard to the limitation recited as 'propagating ... determining system dependency information', the use of Srivastava_2 in the rationale of obviousness will stand. In light of the reply set forth against the arguments regarding claims 21, 25, 29, the Applicants' argument in regard to the propagating steps remain insufficient to overcome the current use of Srivastava (Srivastava_2) in the rejection.

In whole, the claims as submitted and filed 9/13/07 will stand rejected as set forth in the Office action.

Conclusion

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Tuan A Vu
Patent Examiner,
Art Unit 2193
September 30, 2007